In the beginning

In the begin a program was only a logical sequence, Then gosh said: we can’t wait forever, we need to put time on this,

Since then we have two problems: The logical correctness, and the timing correctness.
In theory...

The systems defined as a set of tasks $\tau$
Each task is a set of variables that defines its timing behavior, e.g.,

$$ \tau_i = \{ P, C, D, B, J \} $$

Then, they try to define/develop a scheduler in such way that, for each task $i$ in $\tau$:
the response time of $\tau_i < D_i$
For task level fixed priority scheduler:

\[ \forall \text{ task } i \in \tau: \]

\[ W_i = C_i + B_i + \sum_{j \in hp(i)} \left( \frac{W_i + J_j}{P_j} \right) C_j \]

\[ R_i = W_i + J_i \]

is schedulable \iff \forall \text{ task } i \in \tau | R_i < D_i \]
New metrics for the PREEMPT RT
PREEMPT_RT Timing correctness

- The preempt RT main metric is the latency
  - It is good, per carità...
- But it is very simplistic, if compared to response time.
- Latency is not even clearly defined
  - Kernel is seeing as a black box
  - There is no guarantee that the latency that took place now, will take place in the future (reproducibility/repeatability).
- It very hard, if not impossible, to give any guarantee in those numbers
- We tried to use Extreme Value Analysis – it does not fit in the method.
PREEMPT_RT Timing correctness

- User applications also depend on other characteristics of the kernel:
  - Locking
  - Dependence of other tasks
  - Interference of other tasks (and IRQs)
New metrics for the PREEMPT RT

- How can we improve the situation for Linux?
- What are tasks on Linux?
- What are the other metrics?
  - Execution time of task?
  - Blocking time? (SCHED_STATS)
    - Chain of locks that a task depends
  - Activation delay? (WAKEUP_DELAY)
    - Atomic context delay?
  - Dependency among tasks?
New metrics for the PREEMPT RT

\[ R_i \]

\[ W_i \]

\[ SC_i \]

\[ B_i \]

\[ J_i \]

\[ C_i \]
What will I do, e.g., Composition of Latency
Rescheduling delay

- [need_resched...sched_return]
- Case one: in the schedule
Rescheduling delay

- [need_resched...sched_return]
- Case two: calling the scheduler
- Consider also that we have interference from interrupts
Thoughts?

- It is not reasonable doing this only in user-space
  - Too much data
- Should I do a trace-plugin?
- Use eBPF?
- Do something in kernel (lock stat like?)